5-2016

# Comparison of Karnin Sensitivity and Principal Component Analysis in Reducing Input Dimensionality

Matthew Robert Wilson
*Clemson University*, mwilso9@g.clemson.edu

# Comparison of Karnin Sensitivity and Principal Component Analysis in Reducing Input Dimensionality

---

A Thesis
Presented to
the Graduate School of
Clemson University

---

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Engineering

---

by
Matthew Robert Wilson
May 2016

---

Accepted by:
Dr. Robert Schalkoff, Committee Chair
Dr. Harlan B. Russell
Dr. Yongqiang Wang

# Abstract

Reducing the input dimensionality of large datasets for subsequent processing will allow the process to become less computationally complex and expensive. This thesis tests if Karnin sensitivity can be applied to reducing input dimensions of feed forward neural networks as well as comparing the results to the well known principal component analysis (PCA). The resulting error when reducing dimensions of inputs of various scenarios according to PCA and Karnin sensitivities are compared. After testing, Karnin was found to be able to be used to reduce input dimensions and did as well if not better than PCA in most cases. However, Karnin, like PCA, is not without its weaknesses. To cover both techniques' weaknesses, a combination of the two techniques is introduced. In the end, "PCA chases variance while Karnin chases good mapping."

# Acknowledgments

I would like to thank my advisor and committee chair Dr. Robert Schalkoff for gifting me this topic and helping me throughout it all. I would like to thank him and Dr. Harlan Russell for allowing me to grade for their classes during my graduate schooling so I can save money as well. I would also like to thank Dr. Yongqiang Wang for agreeing to be a member on my committee after my previous member retired. Lastly, I would like to thank Dr. Schalkoff and my parents for their help and understanding while I was hospitalised and recovering.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

In the past, small datasets were used and studied in data analysis. Nowadays, more and more fields are facing larger and larger datasets due to the increase in technology increasing the number of observable variables. As the number of variables become larger, the past methods no longer work. Reducing the dimension of inputs will allow the past methods to work once again. Also, with large datasets with large dimensions, the processing becomes computationally complex and expensive. So reducing the dimensions will allow these to decrease as well [2]. Two techniques to do this are discussed in this thesis. One is the commonly used principal component analysis and the other is Karnin sensitivity, has yet to be applied to this problem.

## 1.1 Principal Component Analysis

Principal component analysis (PCA) is a well known dimension reduction technique [5]. It reduces the data by creating a linear combination of the input data with the largest variance. To do this takes some steps. Matrix $X_{d\times m}$ contains the inputs, where $d$ is the number of inputs and $m$ is the number of samples.

$$X = \begin{bmatrix} \underline{x}_1 \\ \vdots \\ \underline{x}_d \end{bmatrix} \tag{1.1}$$

where $\underline{x}_i$ is a row vector of samples for input $i$. To linearly transform from $X$ to $Y_{d\times m}$, there is matrix $A_{d\times d}$ so that

$$Y = A^T X \tag{1.2}$$

and $A$ is invertible so that

$$X = AY \tag{1.3}$$

Force the expected value of $X$, $E(X)$, to 0 where

$$E(X) = \underline{\mu} = \begin{bmatrix} E(\underline{x}_1) \\ \vdots \\ E(\underline{x}_d) \end{bmatrix} \tag{1.4}$$

This also allows $E(Y) = 0$. This is to cause

$$\Sigma_X = E[(X - E(X))(X - E(X))^T] = E[(X - 0)(X - 0)^T] = E(XX^T) \tag{1.5}$$

where the covariance of $X$ is $\Sigma_X$. Therefore

$$\begin{aligned} \Sigma_Y &= E(YY^T) \\ \Sigma_Y &= E(A^T XX^T A) \\ \Sigma_Y &= A^T E(XX^T)A \\ \Sigma_Y &= A^T \Sigma_X A \end{aligned} \tag{1.6}$$

Now choose $A = M$ so that

$$\Sigma_Y = M^T \Sigma_X M = \Lambda \tag{1.7}$$

where $M$ is the matrix of eigenvectors and $\Lambda$ is the diagonal matrix of corresponding eigenvalues. Now that this eigenvalue matrix equals this covariance matrix, the eigenvalues equate to variances Doing this makes the off-diagonal, the covariances of the principal components, 0 which uncorrolates the components. Now sort $M$ and $\Lambda$ so that

$$M = [A_k : A_D] \tag{1.8}$$

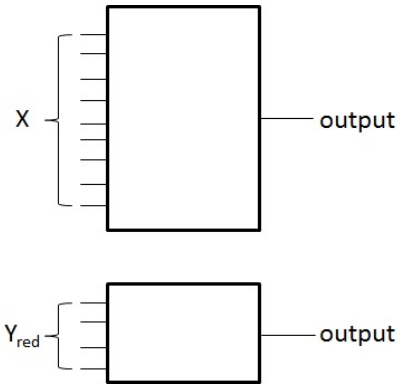



Figure 1.1: Result of PCA

where $A_k$ corresponds to the $k$ largest eigenvalues and $A_D$ corresponds to the $D$ smallest eigenvalues. $D$ represents the number to remove and $k$ represents the number to save. Lastly remove $A_D$ so that

$$Y_{red} = {A_k}^T X \tag{1.9}$$

where $Y_{red}$ of size $k \times m$ and $A_k$ is $k \times k$. Now $Y_{red}$ has $k$ orthogonal rows of the principal components with the largest variances. $X$ has been reduced to $Y_{red}$ while retaining the most important information.

## 1.2 Karnin Sensitivity

Karnin sensitivity is used for sensitivity based pruning [6]. Sensitivity, in this case, refers to an increase in error when removing a weight in the artificial neural network (ANN); the higher the sensitivity, the higher the error. If the sensitivity is zero, there is no difference in error whether the weight is there or not. Figure 1.2 shows an example ANN; each circle is a node and each line is a weight, excluding the lines connecting the input and output to their respective layer. The sensitivity, $S_{ij}$, of weight, $w_{ij}$, can be calculated by this equation:

$$S_{ij} = E(w_{ij} = 0) - E(w_{ij} = w_{ij}^f) \tag{1.10}$$

which means the difference in error when weight $w_{ij}$ is removed (set to 0) and when set to the weight found after training. To effectively prune a node, all the input weights' sensitivities or output weights' sensitivities of that node need to be approximately zero. This will allow the node to

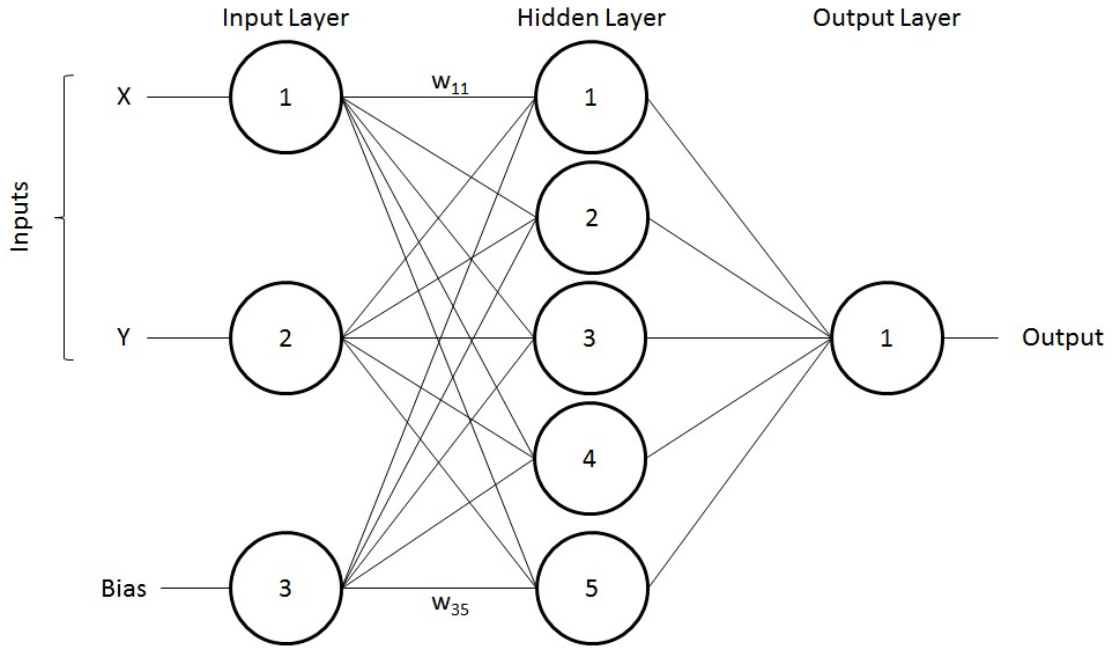



Figure 1.2: Example of a Simple Feed Forward Neural Network

be removed without affecting the error. This is traditionally done in the hidden layer(s) to decrease the complexity of the ANN and reduce the computational expenses.

Another way to calculate this sensitivity, found by Karnin, is during the training of the ANN. This uses the change in weight during each iteration of training instead of after training is finished. After some derivations using gradient descent, the sensitivity can be approximated by:

$$\hat{S}_{ij} = \left[ \sum_{n=0}^{N-1} \frac{(\Delta w_{ij}(n))^2}{\epsilon} \right] \left[ \frac{w_{ij}^f}{w_{ij}^f - w_{ij}^i} \right] \tag{1.11}$$

where $N$ is the total number of iterations, $n$ is the current iteration, and $\epsilon$ is the learning rate. The first half of the equation is calculated during training and the other half is calculated after. Sensitivities should not be negative. If they are, it is because the initial weights were not small enough. Do note that this equation can only handle a flat learning rate. To use momentum or a varying learning rate, the equation will have to be modified.

## 1.3 Research Questions

- Since Karnin sensitivity can be used to prune nodes within an ANN, can it be used to prune inputs as well?
- How does Karnin compare to PCA in reducing input dimensionality?
- As PCA is well investigated, does Karnin succeed where PCA fails?
- Does Karnin sensitivity have its own weakness(es) as well?

# Chapter 2

# Research Design and Methods

## 2.1 Importance of the Study

Because of the large data sets being used nowadays, dimension reduction techniques are being investigated and developed. As feed forward neural networks have high complexity and are computationally expensive, reducing the inputs to the ANNs will lower these. Karnin sensitivity used elsewhere will be significantly expensive because of the need to create a neural network. However, if a neural network will be used anyways, Karnin, if it can be used, will surely be able to help.

## 2.2 Purpose of the Study

The purpose of this study is to establish if Karnin sensitivity can be used to reduce input dimensionality and, if it can, how does it compare to the widely-used PCA.

## 2.3 Explanation of Research Design

This study is of some examples covering various cases and compare the two techniques. During the experiments, it was found that high correlation may lead to problems with Karnin sensitivity. Because of this and that PCA can uncorrelate data, we proposed that combining the two to achieve better results. As such, this third technique was added to the comparison. Karnin with PCA is done by using PCA to uncorrelate the data without reducing dimensions; this is

essentially a transformation of the coordinate system.

## 2.4 Summary of Methods

The appendices contain parts of the MATLAB codes used to create and test these cases. Note that in Section 1.1, the input was described as the rows are inputs and the columns are samples. However, the code had the transpose of that as such Equation 1.9 becomes

$$Y_{red}{}^T = (A_k{}^T X)^T = X^T A_k \tag{2.1}$$

Appendix A shows the MATLAB code for computing PCA of input $in$ of dimension $d$ which is reduced to $reduceTo$ dimensions. Appendix B shows the MATLAB code for training the ANN and calculating the sensitivity of its weights. The initial values of many variables are subject to change in order to train the data set.

Cases using Karnin sensitivity were run by training the ANN and calculating the corresponding sensitivity. After that, the weights attached to inputs were removed based on sensitivity and then the ANN was re-run to test for error. Cases using PCA were run by reducing dimensions by PCA and then train the ANN with this new data. After which, the ANN was run to test for error. Cases using Karnin with PCA, as with the name, is a combination of the two. This was done to cover the failures of the two individual techniques discussed later. First, PCA was run without reducing dimensions. Next, the ANN was trained and sensitivities were calculated. Lastly, after removing input weights based on sensitivity, the ANN was run to test for error. Every ANN had a bias and $2d + 1$ hidden layer nodes. The number of input and output nodes are dependent upon the problem. Each case, except for Case 1 and Case 8, had a sample size of 10,000. Case 1 had a sample size of 16 as that could completely cover all possibilities. Case 8 had a sample size of 166, which was equal amounts of both classes.

# Chapter 3

# Related Work

There has been much research in reducing input dimensions. However, Karnin sensitivity has not been used for this. Karnin sensitivity can be used to prune insensitive weights and nodes of a feed-forward neural network. Why not use this to prune inputs instead of nodes? Because there has not been much research involving Karnin, for more information take a look at his original work [4] . This thesis also introduces the combination, Karnin sensitivity with PCA, to this problem.

Although Karnin has not been used to reduce input dimensions, there are many other techniques usable. One of which is principal component analysis. Kambhatla and Leen [3] discuss specifically PCA techniques in depth. For other techniques, [1] and [2] explain many various dimension reduction techniques.

# Chapter 4

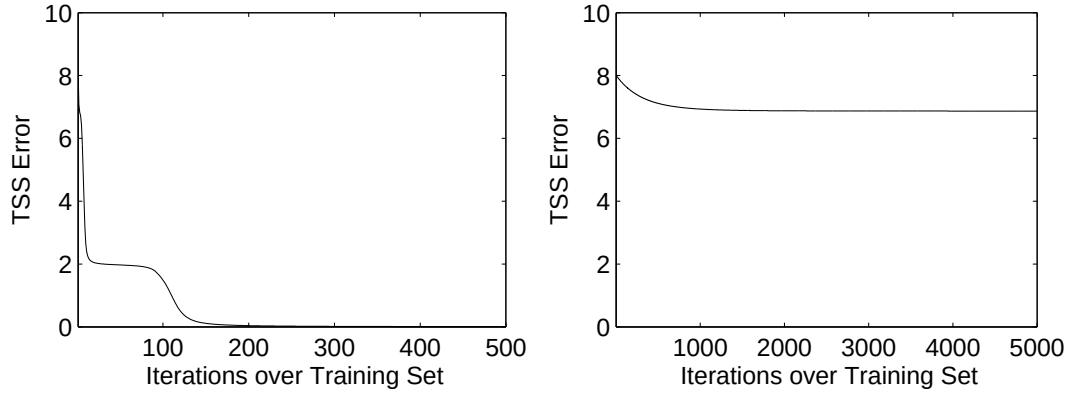
# Results

## 4.1 Case 1: Binary Input

This case depicts various binary operations, with binary inputs and outputs. The training set data contains the complete binary input and output, all possible cases for a specific binary operation. Karnin sensitivity, PCA, and Karnin with PCA were used to determine which binary inputs are necessary for the output. These were four inputs that were either on (1) and off (0). The binary expression for one particular training set would be $AB + \bar{A}\bar{B}\bar{C}\bar{D}$. Because this uses an ANN, the binary values were not 1 and 0 but 1 and $-1$. This allows the ANN to be able to be trained. The case is to reduce this four dimensional input to two dimensions with as little error as possible.

### 4.1.1 Karnin

Figure 4.1a shows the TSS error for the training of the ANN for Karnin. TSS Error is the total sum squared error, the overall sum of the squared error. Using four inputs, this data set could be trained well. The Karnin sensitivity calculated during this training was $\begin{bmatrix} 0.9327 & 0.9030 & 0.1252 & 0.1245 \end{bmatrix}$. This shows that the first two inputs, A and B, are more important than inputs C and D. Removing these two inputs (setting their respective weights to zero) has shown an increase in error because $\bar{A}\bar{B}\bar{C}\bar{D}$ no longer has an output of 1. This is the only difference after removing these two inputs.




(a) Case 1 Karnin Sensitivity Training Error over Iterations

(b) Case 1 PCA Training Error over Iterations

Figure 4.1: Case 1 Karnin and PCA Training TSS Error

### 4.1.2 PCA

PCA did not work well in this case. This is because the variance of the inputs are all equal. The reduction matrix in this case is the identity matrix. This causes the reduced matrix to pick randomly two of the four inputs and remove the other two. The program used preserved inputs C and D, which caused the subsequent training to fail, as shown in Figure 4.1b.

### 4.1.3 Karnin with PCA

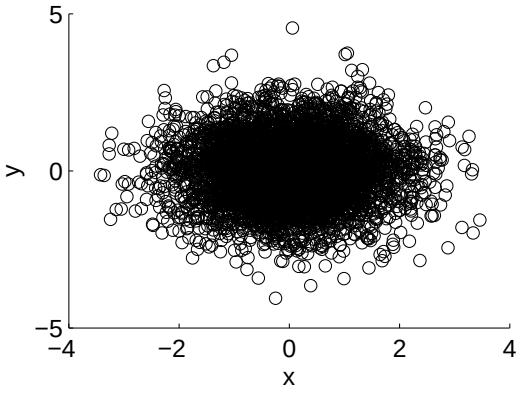
Because the reduction matrix in this case is the identity matrix, there is no difference in this case between Karnin and Karnin with PCA.

## 4.2 Case 2: Irrelevant Inputs

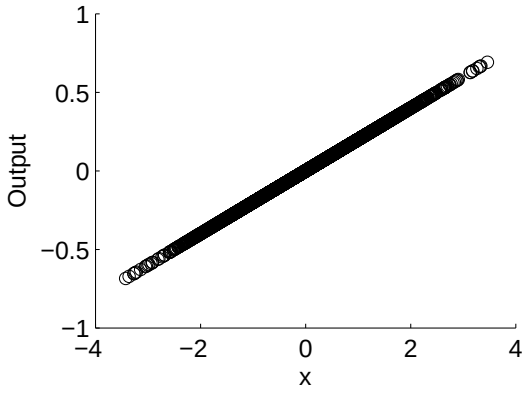
This case considers the situation where the output is a direct function of one of multiple inputs. This example depicts $F(x, y) = \frac{x}{5}$, where $x$ and $y$ are random Gaussian variables around 0 with a standard deviation of 1.

### 4.2.1 Karnin

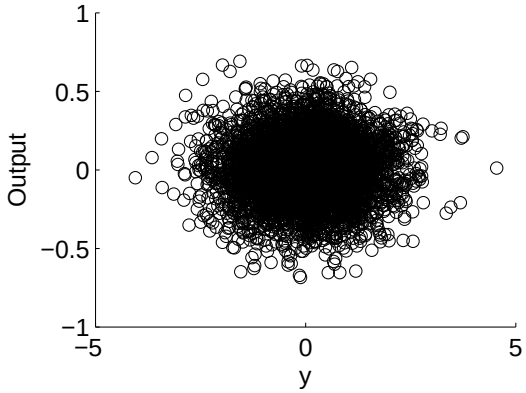
The sensitivity found after training this equation was $\begin{bmatrix} 46.0085 & 0.0004 \end{bmatrix}$ for inputs $x$ and $y$. This shows that Karnin found $x$ to be important and $y$ to be irrelevant. Thus removing $y$ and




(a) X vs Y




(b) X vs Output




(c) Y vs Output

Figure 4.2: Case 2 Relationship between Inputs and Outputs

recalculating the error from the training set, the total error raised from 0.3721 to 0.3722.

### 4.2.2 PCA

After reducing the number of inputs to one using PCA and training the transformed data, the error came out to be 99.4664. This is because the transformation matrix used by PCA to reduce the dimensions was $\begin{bmatrix} 0.0479 \\ 0.9989 \end{bmatrix}$. This shows that PCA found input $y$ to be more important and preserved it. This is because for this particular input data set, the covariance was $\begin{bmatrix} 0.9949 & 0.0011 \\ 0.0011 & 1.0170 \end{bmatrix}$. PCA wants to preserve high variance and thus chose the input $y$ over input $x$ because it had slightly more variance. This shows that PCA is not good for direct relationship situations, where the output is a mathematical equation of the input. This is because PCA does not take the output into account and only deals with the input.

### 4.2.3 Karnin with PCA

The sensitivity found after training this equation was $\begin{bmatrix} 47.8936 & 0.1126 \end{bmatrix}$ for inputs $x$ and $y$. This shows that Karnin found $x$ to be important and $y$ to be irrelevant. Thus removing $y$ and recalculating the error from the training set, the total error raised from 0.3316 to 0.5589. There is barely any difference between this and just Karnin alone.

## 4.3 Cases 3-5: Interscatter Distance versus Intrascatter Distance

Interscatter distance, $S_b$, is the distance between two data classes while intrascatter distance, $S_w$, is the distance within the classes. These affect the interclass and intraclass variances, respectively. Because this is related to variance, the following cases are important for PCA. As PCA places importance on the larger variances, the following cases are examples of when one of the distances is larger than the other. This instance is also where PCA largely fails.

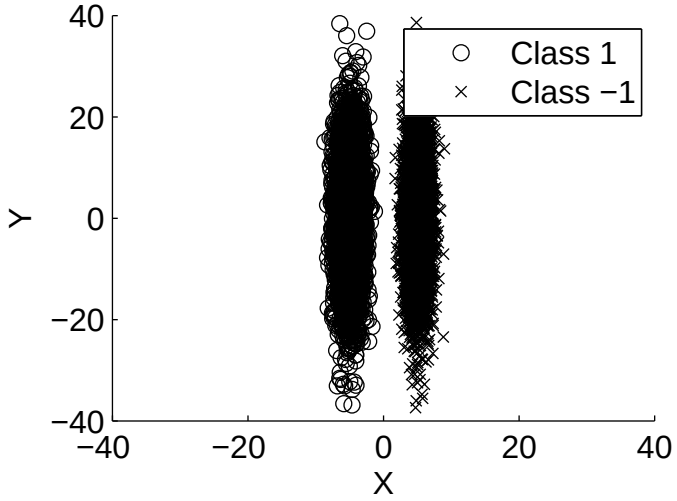



Figure 4.3: Case 3 Inputs

### 4.3.1 Case 3: Intraclass Variance > Interclass Variance, Failure of Principle Component Analysis

PCA focuses on the variances of the data. It places high importance on the higher variances. Therefore, if an unnecessary input has higher variance than the necessary input, PCA will fail. This case is one such case. The data used is shown in Figure 4.3.

#### 4.3.1.1 PCA

The variance of input $x$ was found to be 26.0832, while input $y$ was 100.5493. Because PCA gives priority to higher variance, it will incorrectly preserve input $y$. After using PCA and attempting to train the ANN, the error was found to be 4998.7000.

#### 4.3.1.2 Karnin

Karnin was not fooled like PCA was. The sensitivity was 6408.8 for input $x$ and 3.2738 for $y$. This shows that $x$ is much more needed than $y$ when training this data set. Both before and after removing input $y$ from this ANN, the error was 0.1667. This means that Karnin chose correctly and that remove input $y$ had no affect on the ANN.

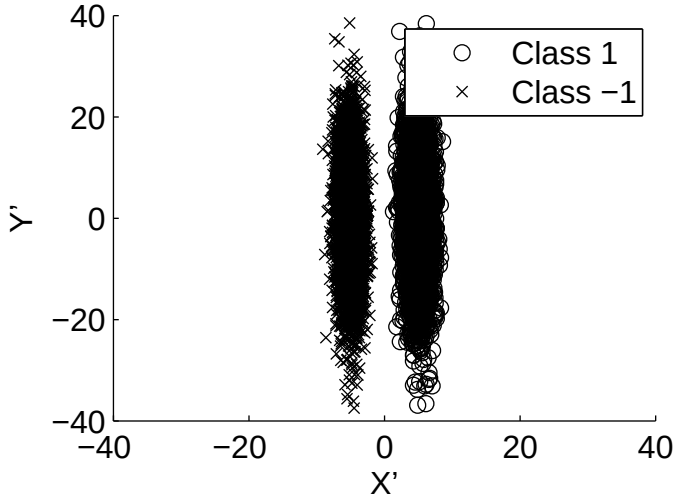



Figure 4.4: Case 3 Transformed Inputs

#### 4.3.1.3 Karnin with PCA

PCA transformed the input, as seen in Figure 4.3, to the inputs seen in Figure 4.4. After training the ANN with this transformed data set, the sensitivity of $x'$ and $y'$ was found to be 3370.4 and 2.5202 respectively. This shows that input $x'$ of the transformed data set has more importance to the ANN the input $y'$. The error after training and after removing input $y'$ was 1.5863 and 1.5918, respectively.

### 4.3.2 Case 4: $S_b < S_w$

This case can take multiple examples one such example is Case 3 described in Section 4.3.1. $S_w = \begin{bmatrix} 2.0631 & 0.1138 \\ 0.1138 & 195.9684 \end{bmatrix}$ and $S_b = \begin{bmatrix} 50.1145 & -1.2201 \\ -1.2201 & 0.0297 \end{bmatrix}$. Another example is shown in Figure 4.5. In this case $S_b = \begin{bmatrix} 6.1346 & 13.1454 \\ 13.1454 & 28.1687 \end{bmatrix}$ while $S_w = \begin{bmatrix} 44.5537 & -16.3128 \\ -16.3128 & 8.2544 \end{bmatrix}$.

#### 4.3.2.1 Karnin

The sensitivities for this example are 304.1 and 1944.2. This means that although much less than input $y$, input $x$ have enough importance to not be removed. When removing input $x$, the error greatly increased from 0.4798 to 1187.7. While this example fails according to the resulting error, it actually succeeds because it removes the least sensitive input which results in the lowest

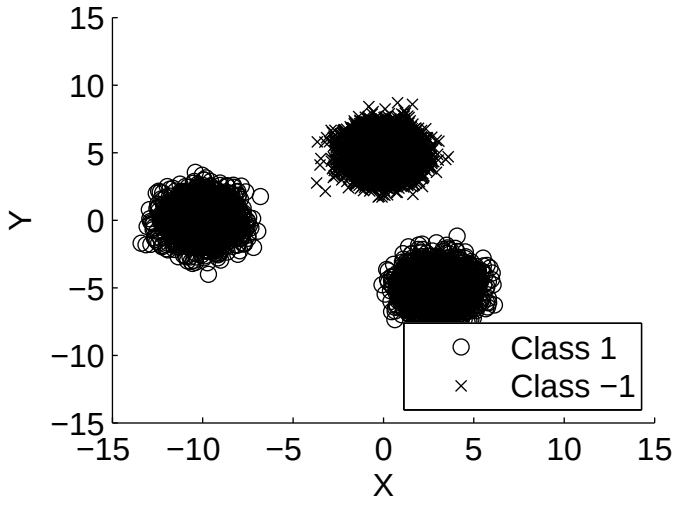



Figure 4.5: Case 4 Inputs where $S_b < S_w$

possible error when directly removing one input. If instead of $x$, $y$ was removed, the error would have shot up to 9605.6. As Karnin successfully reduces the inputs in Case 3 in Section 4.3.1, the reason for failure is not because of the interscatter and intrascatter distances. It is only because both inputs are sensitive, or relevant, in this case that the error increased so much.

#### 4.3.2.2 PCA

As with the case described in Section 4.3.1, this instance failed as well.The transformation matrix for the inputs was $\begin{bmatrix} -0.9780 \\ 0.2086 \end{bmatrix}$ showing that PCA incorrectly thinks that input $x$ hold more importance that input $y$. The error for this example was 53.3567. As variance and the two distances are related, the inputs that reflect higher values in the covariance matrix will have higher distance (inter or intra). In this case the highest value is with input $x$ on $S_w$. This is reflected in the covariance matrix where the largest variance is also with input $x$.

#### 4.3.2.3 Karnin with PCA

Figure 4.6 shows the transformed inputs calculated by PCA without reducing dimensions. After using this new data, the ANN was trained. The sensitivity calculated during this training was 2586.4 and 53.3210 for inputs $x'$ and $y'$. The error before and after removing the least important input, input $x'$, was 0.3802 and 8.6166 respectively. This shows that it found the correct solution,

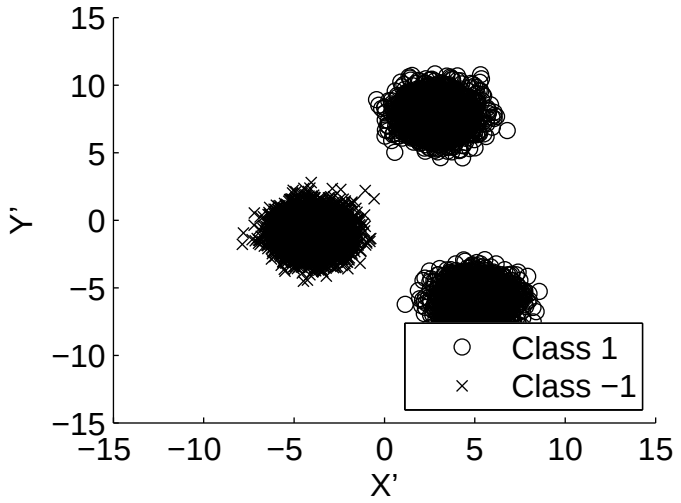



Figure 4.6: Case 4 Transformed Inputs

much better than either PCA or Karnin found.

### 4.3.3 Case 5: $S_b > S_w$

An example of this case has $S_w = \begin{bmatrix} 14.4298 & 1.8003 \\ 1.8003 & 2.2503 \end{bmatrix}$ and $S_b = \begin{bmatrix} 45.0331 & 7.0176 \\ 7.0176 & 1.0936 \end{bmatrix}$ which is shown in Figure 4.7.

#### 4.3.3.1 Karnin

The sensitivities for this example are 2273.4 and 81.8451. This means that input $y$ has much less effect on error than input $x$ and thus can be removed. When removing input $y$, the error increased from 17.6678 to 18.3401. This shows that Karnin found the correct input to remove.

#### 4.3.3.2 PCA

The transformation matrix for the inputs was $\begin{bmatrix} -0.9780 \\ 0.2086 \end{bmatrix}$ showing that PCA incorrectly thinks that input $x$ hold more importance that input $y$. The error for this example was 18.8851. This shows that PCA works best when $S_b > S_w$.

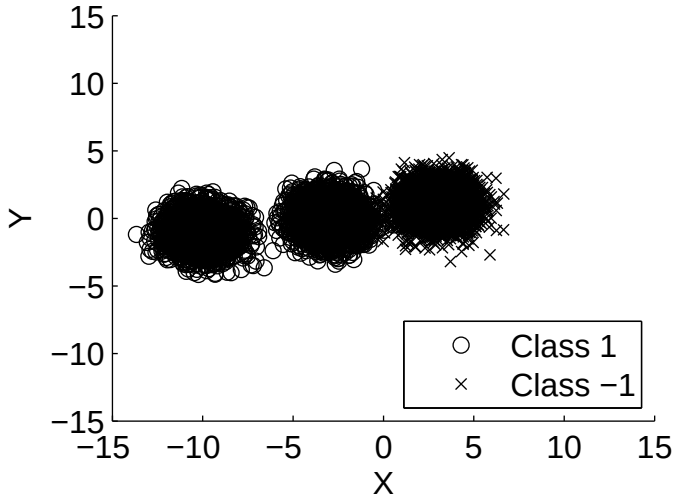



Figure 4.7: Case 5 Inputs where $S_b > S_w$

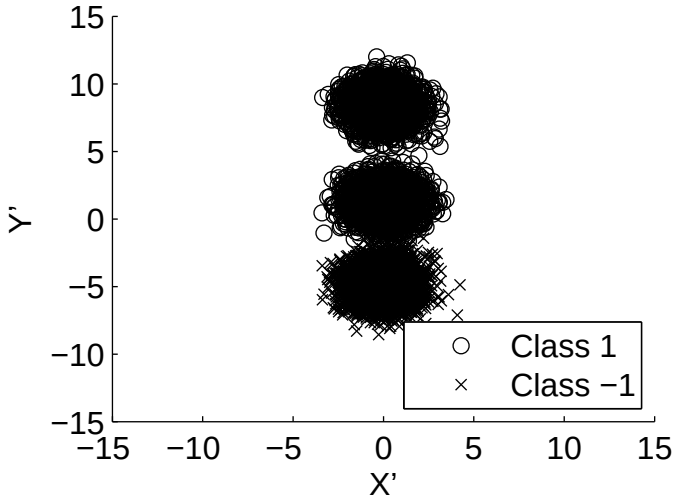



Figure 4.8: Case 5 Transformed Inputs

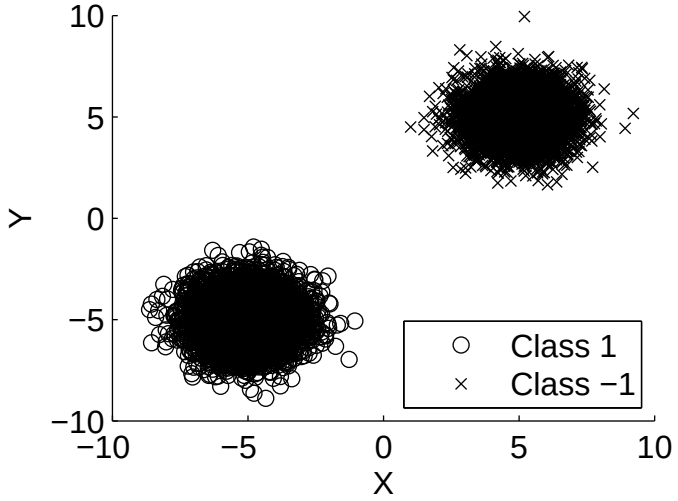



Figure 4.9: Case 6 Inputs

### 4.3.4 Karnin with PCA

Figure 4.8 shows the transformed inputs calculated by PCA without reducing dimensions. After using this new data, the ANN was trained. The sensitivity calculated during this training was 0.2463 and 2038.4 for inputs $x'$ and $y'$. The error before and after removing the least important input, input $x'$, was 17.9420 and 18.0900 respectively. This shows that it found the correct solution, much better than either PCA or Karnin found.

## 4.4 Cases 6-7: High Correlation

The following cases are important for Karnin sensitivity as they can cause errors to appear. This is because Karnin does not change the inputs themselves, only directly removing one or more of them. With high correlation, the inputs are related to one another. This means that if one input is sensitive, or relevant, then it means that other inputs that are highly correlated will also be sensitive even though they may just be repeated information.

### 4.4.1 Case 6: Highly Correlated Inputs, Failure of Karnin Sensitivity

As PCA has its failures, Karnin does as well. One problem occurs when more than one input has approximately equal sensitivity. This means that these inputs have equal importance to the ANN. However this can mean either that all the involved inputs are needed or only one is

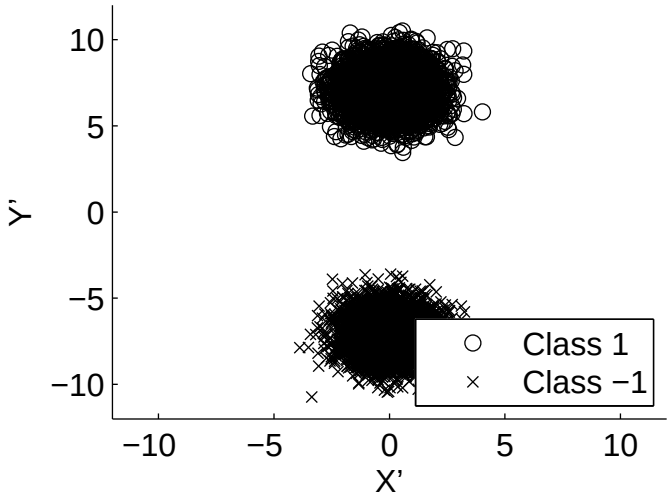



Figure 4.10: Case 6 Transformed Inputs

needed, though it does not matter which one. An example of the latter is used here and shown in Figure 4.9. It can be seen that either input, $x$ or $y$, could be used by itself to classify the two inputs. The correlation matrix for this case is $\begin{bmatrix} 1.0000 & 0.9609 \\ 0.9609 & 1.0000 \end{bmatrix}$.

#### 4.4.1.1 Karnin

The sensitivity for $x$ and $y$ are 841.5041 and 840.8277, respectively, showing equal importance for the two inputs. As such, when removing one input, in this case $y$, the error increased from 2.8975 to 26.5844. This is not the expected large increase in error when removing an input with a large sensitivity. This shows that Karnin sensitivity is not always reliable. While this did not cause a large increase in error, Karnin still does worse than the other two techniques.

#### 4.4.1.2 PCA

The total error after training the PCA-transformed data set was 0.4274. This means that PCA was not led astray and successfully reduced dimensions and trained the ANN.

#### 4.4.1.3 Karnin with PCA

Figure 4.10 shows the transformed inputs calculated by PCA without reducing dimensions. After using this new data, the ANN was trained. The sensitivity calculated during this training was

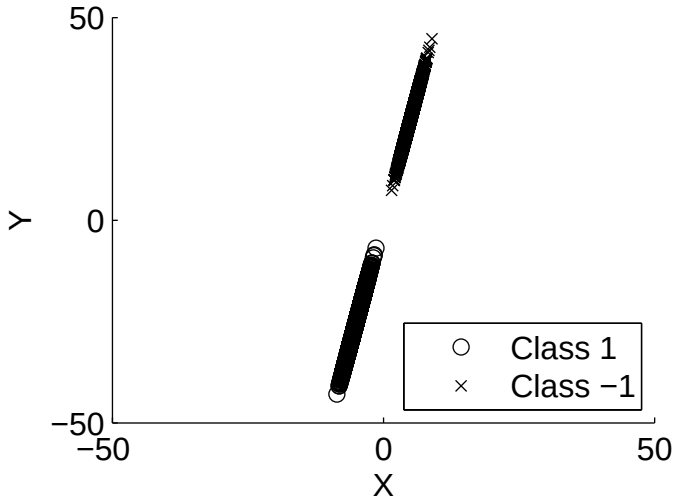



Figure 4.11: Case 7 Inputs

0.0003 and 1496.5 for inputs $x'$ and $y'$. The error before and after removing the least important input, input $x'$, was both 2.9400. This is because PCA uncorrelates the data and then Karnin removes the irrelevant inputs.

### 4.4.2 Case 7: Directly Related Inputs

The relationship does not need to be 1:1 to be highly correlated. Another example is shown in Figure 4.11. Additionally, these inputs are directly related: $Y = 5 \times X$. As these inputs are directly related, the correlation matrix is $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$.

#### 4.4.2.1 Karnin

While Karnin successfully reduces the dimension, as with Case 6 of Section 4.4.1, the sensitivity does not reflect the error. The sensitivity was found to be $\begin{bmatrix} 780 & 1950.4 \end{bmatrix}$ for inputs $x$ and $y$. The error increased from 0.1694 to 0.1723 after removing $x$. As can be seen, there was hardly any error before and after the pruning. This shows that the sensitivity of input $x$ was erroneous.

#### 4.4.2.2 PCA

The error was found to be 4.4574 after using PCA and training the ANN. While it is slightly worse, there is no effective difference between PCA and Karnin as far as the results. It reduced the

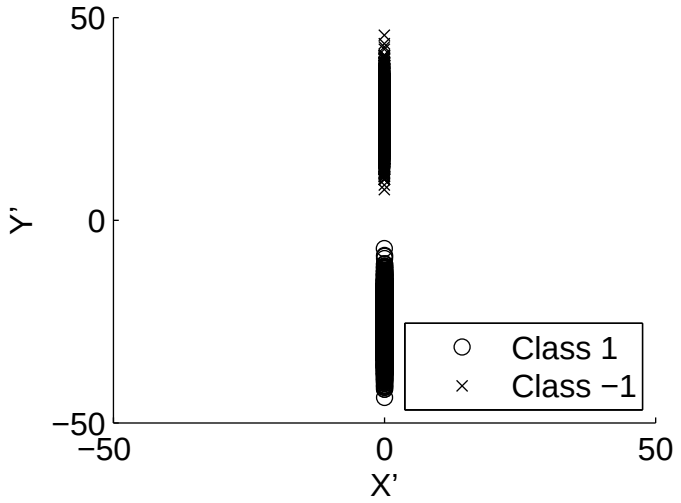



Figure 4.12: Case 7 Transformed Inputs

dimensions and trained well.

#### 4.4.2.3 Karnin with PCA

Though it may be hard to see in Figure 4.12, all $x'$ values of the transformed inputs are 0. This is because when using PCA to change the coordinate system of directly related inputs, it removes the one with less variance. Though there is no need to use Karnin sensitivity since one of the inputs are already 0, the sensitivities for inputs $x'$ and $y'$ were $\begin{bmatrix} NaN & 30154 \end{bmatrix}$. Because input $x'$ was always 0 for each of the data points in the training set, the weights attached to input $x'$ never changed. According to Equation 1.11 on Page 4, when the final weight minus the initial weight is 0, the equation divides by 0 which cannot be done, thus $NaN$. There was no change in the error, 0.0166, when removing input $x'$.

## 4.5 Case 8: High Dimension Inputs

This case uses real world electroencephalogram (EEG) data for schizophrenia. The data has 27 inputs and 2 classes (possible outputs). Figure 4.13 shows the total sum squared error when the neural network reduces input dimension from the 27 original inputs to 1 to 27 inputs.

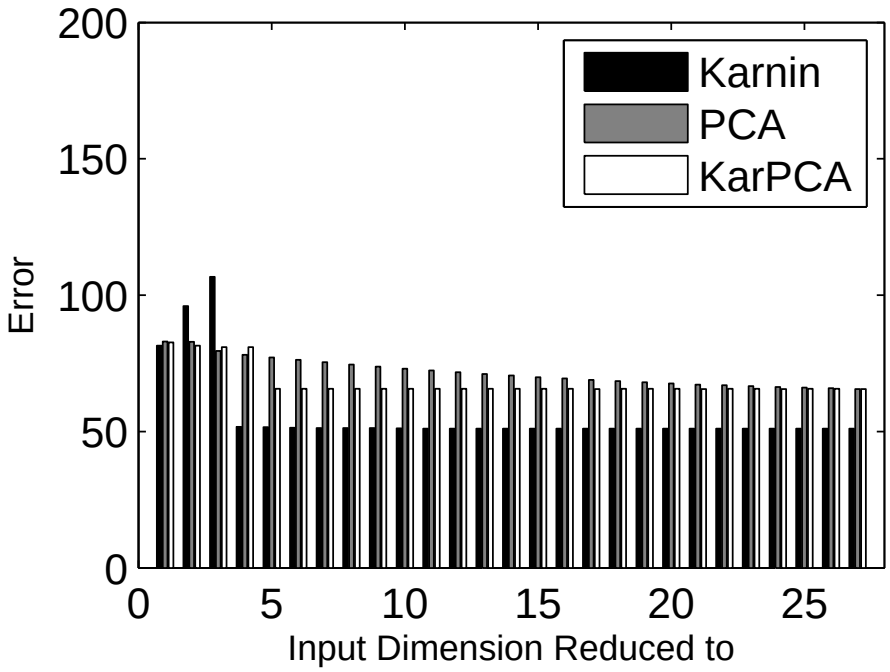



Figure 4.13: Case 8 Error for True Output of each Technique

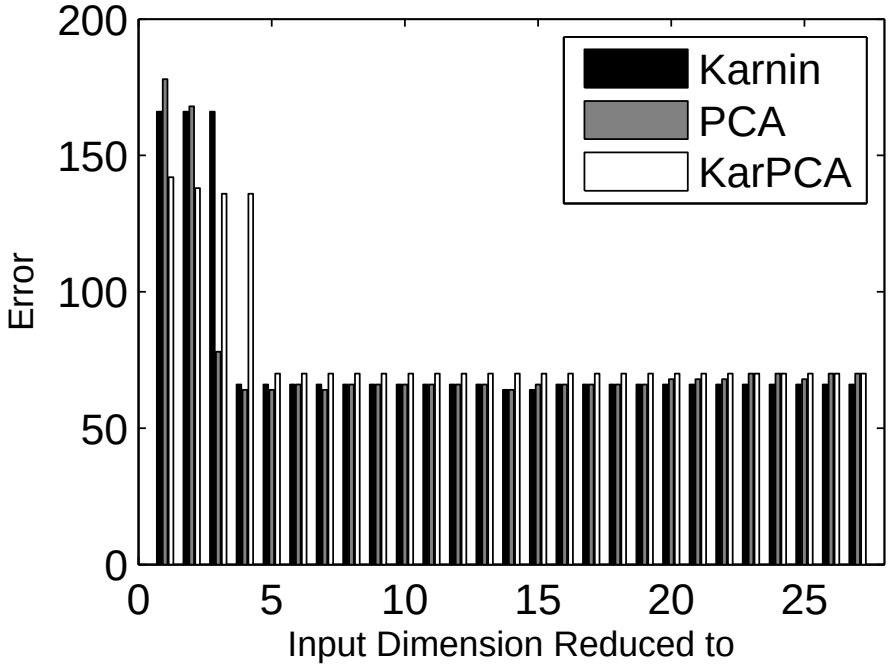



Figure 4.14: Case 8 Error for Rounded Output of each Technique

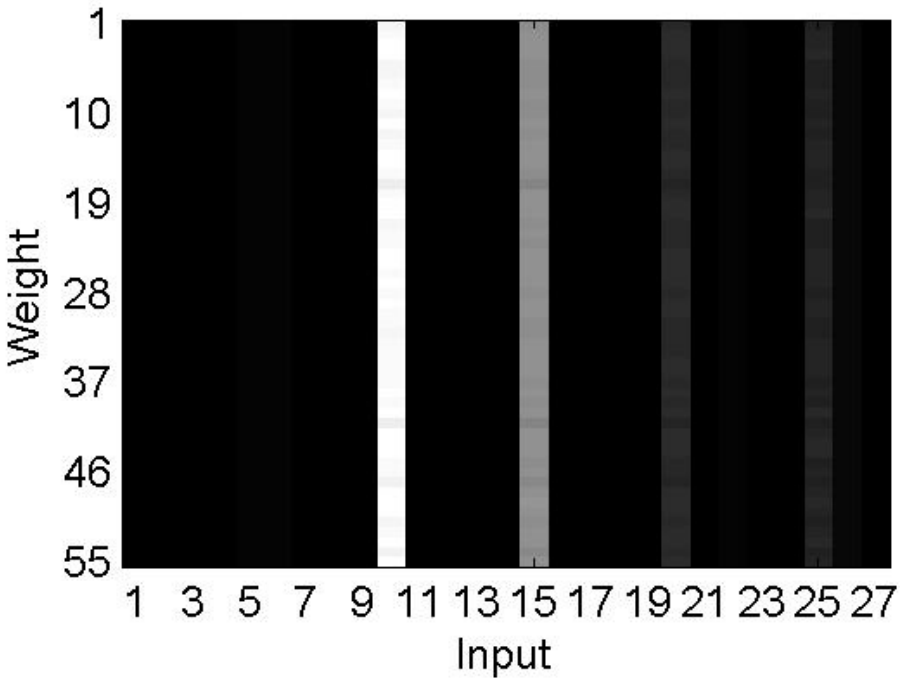



Figure 4.15: Case 8 Karnin Sensitivity of Inputs

### 4.5.1 Karnin

It can be seen in Figure 4.13 that the error does not change until the network is reduced to three inputs. This is because there are four inputs with high sensitivity while the others are irrelevant. Figure 4.15 shows the sensitivity of the inputs. The whiter the cell, the larger the sensitivity. The x-axis represents the inputs while the y-axis represents the weights between the inputs and the hidden layer nodes. It can be seen that four inputs have much higher sensitivity than the other inputs. The error reduces when below three inputs because the output becomes closer to zero instead of the extremes (1 or -1) so the error will decrease. As can be seen, Karnin shows no difference when removing irrelevant inputs, however, when a relevant input is removed, the network breaks and will not work. High dimensionality will not affect Karnin. When removing irrelevant inputs there will be little to no change in error no matter how many irrelevant inputs removed, yet when removing any relevant input, the network will break.

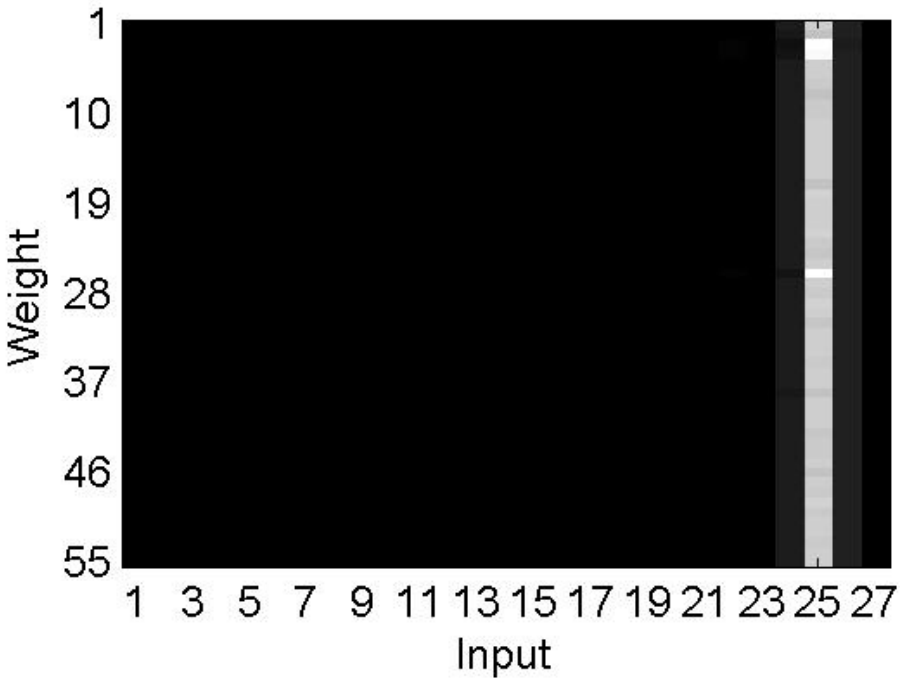



Figure 4.16: Case 8 Karnin Sensitivity of Inputs

### 4.5.2 PCA

Figure 4.13 shows that with each input reduced, there is a slight increase in error. This is because PCA creates a linear combination of the inputs as the new reduced dimension inputs. This means that there will be some irrelevant data kept in the network. Therefore, with more inputs removed, more irrelevant data will be added to the relevant inputs which creates more error in the output. However, compared to Karnin, PCA does a much better job of reducing inputs, relevant and irrelevant. There is only a slight increase of error when reducing past the amount of relevant inputs. This is for the same reason as before: PCA creates new inputs as a linear combination of the original inputs. High dimensionality will affect PCA: the more inputs removed, the larger the error will be.

### 4.5.3 Karnin with PCA

As this is a combination of Karnin and PCA, so is the resulting error. Like Karnin, the error did not change until the network broke by removing a relevant input. Like PCA, there is only a slight increase in error when removing more than one relevant input. Because the new uncorrelated

| *Case* | *Karnin* | *PCA* | *Karnin w/ PCA* |
|---|---|---|---|
| **1 Binary** | 1.9782 | 13.7474 | 1.9751 |
| **2 Irrelevant** | 0.3722 | 99.4664 | 0.5589 |
| **3 PCA Fail** | 0.1667 | 4998.7000 | 1.5918 |
| **4** $S_b < S_w$ | 1187.7000 | 53.3567 | 8.6166 |
| **5** $S_b > S_w$ | 18.3401 | 18.8851 | 18.0900 |
| **6 High Corr** | 26.5844 | 0.4274 | 2.9400 |
| **7 Dir Rel** | 0.1723 | 4.4574 | 0.0166 |

Table 4.1: Total TSS Error of cases for each Technique

inputs, found by using PCA, irrelevant data is combined with relevant data creating more error overall. Figure 4.16 shows the sensitivity found after training this data set. Though it may be hard to see, there are three sensitivities much larger than the others. However, there are still two more relevant inputs, according to Karnin. It is only because the value of the sensitivities are about a tenth of the largest sensitivities and half of the third largest. This causes it to not be able to be seen in Figure 4.16. Overall, this does worse than Karnin but better than PCA. For the same reasons as with Karnin, high dimensionality will not affect Karnin with PCA. In Figure 4.13, it can be seen that Karnin with PCA starts out with the same error as PCA does, because there was not reduction occurring. When the reduction occurs, it acts as Karnin does: no minimal error change until a relevant input is removed.

There is an error during this case with Karnin with PCA, though the cause is currently unknown. There should be only four relevant inputs, not five. In Figures 4.13 and 4.14, when the reduced dimension was reduced from five to four, there was little change in error. After going back and switching these two inputs order of being removed, it was found that the fourth most sensitive was actually irrelevant and the fifth was relevant.

## 4.6 Conclusion of Results

Overall Karnin sensitivity and PCA both have their own weaknesses. PCA relies on the variance and distances of the inputs and thus fails when the irrelevant inputs have larger variances than the relevant ones, as shown in Sections 4.3.1.1 and 4.3.2.2. Karnin sensitivity will fail, or rather be misrepresented, when input correlation is high as any input removed would increase the error a little in comparison to the expected value as can be seen in Sections 4.4.1.1 and 4.4.2.1.

Because Karnin does not modify the inputs at all, Karnin may find the best solution given the inputs, but not the best possible solution. The orientation may prevent this from happening as shown in Section 4.3.2.1. Karnin with PCA combines the two and covers many of their weaknesses. It uses PCA to uncorrelate the input without reducing the dimension and then uses Karnin to find the least sensitive, or irrelevant, inputs to remove. Most of the time, there will be little difference in results between the two latter methods, however, when it is needed, Karnin with PCA does much better than just Karnin because the uncorrelated inputs. Another weakness that Karnin has and is shared with Karnin with PCA is that for the sensitivity to be correct the data set needs to be sufficiently trained.

# Chapter 5

# Conclusions and Discussion

## 5.1 Answering the Research Questions

- Since Karnin sensitivity can be used to prune nodes within an ANN, can it be used to prune inputs as well?

  Yes. It is most adept at identifying and removing irrelevant inputs.

- How is Karnin compared to PCA in reducing input dimensionality?

  It is better in removing irrelevant inputs yet may not find the best overall solution as PCA does.

- As PCA is well researched, does Karnin succeed where PCA fails?

  Yes. PCA fails when the intraclass variance is larger than interclass variance. Karnin does not fall for this and correctly identifies and removes the irrelevant input. See Case 3 of Section 4.3.1.

- Does Karnin have its own weakness(es) as well?

  *Error* Highly correlated inputs may lead to a misrepresentation in sensitivity: high sensitivity yet low error when removed.

  *Weakness* Orientation of inputs. May not give the best possible solution due to the constraints of the inputs.

## 5.2 Connections and Conclusions

Overall, Karnin can be used to reduce input dimensions. "PCA chases variance while Karnin chases good mapping." In many cases Karnin does as well as, if not better than, PCA. The only times it does not do as well as PCA is during its weakness or error. Karnin's error is high sensitivity yet low error during highly correlated inputs. And Karnin's weakness is orientation: it finds the best solution without modifying the inputs, not the best possible solution. However Karnin with PCA, which does transform the inputs, does not have these two weaknesses and does not have PCA's weakness as well. Do note that PCA does a good job of preserving the relevant inputs. This can be seen well when using Karnin with PCA as the sensitive inputs were almost always the last ones when running the program (the software sorts the variances from least to greatest).

What Karnin sensitivity is best at is portraying the importance of each input. This way you can know not only what inputs are safe to remove but also which inputs are relevant. For example, in Case 8, the inputs are EEG data for schizophrenia. With Karnin, the relevant inputs can be identified and know which parts of the brain are important to schizophrenia. When reducing a large number of inputs, it is hard to see how preserved each input is in PCA. However, Karnin sensitivity can be found for each input from each weight. Added together and the sensitivity for each input can be easily compared.

## 5.3 Theoretical Implications and Recommendations for Further Research

Karnin can be seen as usable in reducing dimensions and the results are comparable to that of PCA, therefore, in the future, the process of the two should be compared. As the purpose of reducing inputs is to reduce complexity and expenses, the number of computations of Karnin sensitivity's shadow array and PCA's input transformation should be compared. At what number of inputs/reduced inputs will one be less computationally expensive than the other? Add in Karnin with PCA and determine if this increase in computations is worth the security of the result.

During the testing, it was found that PCA took longer (more iterations) to fully train the ANN in most scenarios than the other two techniques. The only exception was Case 8 High Dimension Inputs, but that was only because none of the three techniques could fully train the ANN

within 30,000 iterations. Is this true for all cases? Could it be easier to map with all inputs than it is with reduced inputs? This could affect or be included with the previous question.

Karnin with PCA was introduced since it could cover the weaknesses and failures of the two individual techniques. Thus, it was not extensively studied. Does this technique have any glaring weakness(es)? What led to the error in Case 8?

The cases showcased in this thesis used a hidden layer of $2d + 1$ in the ANN, where $d$ is the number of initial inputs. How effective would it be to train an ANN that had a hidden layer of $2r + 1$, where $r$ is the reduced amount of inputs, from the beginning? Meaning, how effective would it be to train an ANN with the reduction in mind versus one that does not? This would also help reduce the complexity of the ANN.

Karnin sensitivity should be modifiable to allow momentum and/or a varying learning rate. Both of these would help in training the ANN faster and more easily thus reducing some iterations of computations.

# Appendices

# Appendix A PCA Dimension Reduction Program, MATLAB

Referenced on Page 7

```
%% reduce dim w/ principal component analysis
u = mean(in);                    % find the average value of each input
E = zeros(size(in));
for i=1:d                        % for each training data, subtract the average
    E(:,i) = in(:,i) - u(i)*ones(length(in),1);
end
sub = in(1,:) - E(1,:);          % store this value to transform any input
C = cov(E);                      % find covariance
[M,S] = eig(C);                  % find eigenvectors (M) and values (S)
Ak = M(:,(end-reduceTo+1):end);  % reduce dimensions if needed and store for later
                                 % larger eigenvalues the later in the matrix
Yred = E*Ak;                     % determine transformed inputs for training

...

[data_len,~] = size(in);
Yred2 = (in - repmat(sub,data_len,1))*Ak; % transform inputs for using ANN
```

# Appendix B ANN Training and Karnin Sensitivity Calculation Program, MATLAB

Referenced on Page 7

```
%% initialize variables
% set learning rate
p = 0.0001;
% set total iterations for training
iter = 1;
maxIter = 5000;
error = zeros(maxIter,1);
sens_sumh = 0;

%% initialize weights
std = 0.00001;
wh = normrnd(0,std,[2*d+1 d+1]);
[wr,wc] = size(wh);
ih = wh;
wo = normrnd(0,std,[c 2*d+1+1]);
io = wo;

%% train ANN
while iter<=maxIter
    dwo = 0;
    dwh = 0;
    for i = data_len:-1:1
        % run each data point through ANN
        input = in(i,:)';
        % layer 1
        neth = wh*[input;1];
```

```
        oh = tanh(neth);
        % layer 2
        neto = wo*[oh;1];
        oo = tanh(neto);

        % calculate error
        error(iter) = error(iter) + norm(out(i,:)' - oo)^2;

        % change in error with respect to layer weights (done as matrices)
        deltao = ((out(i,:)' - oo).*(1 - oo.^2));
        deltah = (deltao'*wo(:,1:end-1))'.*(1 - oh.^2);
        dwo = dwo-p*deltao*[oh;1]';
        dwh = dwh-p*deltah*[input;1]';
    end
    % apply the change in weight
    wo = wo-dwo;
    wh = wh-dwh;
    % sum total the weight change for sensitivity calculations
    sens_sumh = sens_sumh + dwh.^2/p;
    iter = iter + 1;
end
%% compute sensitivity
sensh = sens_sumh.*(wh./(wh-ih));
sum_sens = sum(sensh);
disp('Sum of Karnin Sensitivities of Inputs');
disp(sum_sens);

...

%% reduce dimensions based on sensitivity
% in case there is a NaN value
```

```
for i=1:d
    if(isnan(sum_sens(i)))
        sum_sens(i) = 0;
    end
end
% sort in increasing sensitivity
[~, index] = sort(sum_sens(1:end-1));
% replace weights attached to input with 0
for i=1:d-reduceTo
    wh(:,index(i)) = zeros(size(wh(:,index(i))));
end
```

# Bibliography


[1] Miguel A Carreira-Perpinán. A review of dimension reduction techniques. *Department of Computer Science. University of Sheffield. Tech. Rep. CS-96-09*, 9:1–69, 1997.

[2] Imola K Fodor. A survey of dimension reduction techniques, 2002.

[3] N. Kambhatla and T. K. Leen. Dimension reduction by local principal component analysis. *Neural Computation*, 9(7):1493–1516, July 1997.

[4] E. D. Karnin. A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, 1(2):239–242, Jun 1990.

[5] Mark Richardson. Principal component analysis. *URL: http://people. maths. ox. ac. uk/richardsonm/SignalProcPCA. pdf (last access: 3.5. 2013). Aleš Hladnik Dr., Ass. Prof., Chair of Information and Graphic Arts Technology, Faculty of Natural Sciences and Engineering, University of Ljubljana, Slovenia ales. hladnik@ ntf. uni-lj. si*, 2009.

[6] R. Schalkoff. *Artificial Neural Networks*. McGraw-Hill Companies, Inc., 1997.